Math 557 Oct 24

Recursive Functions

Primitive recursion

The following functions are certainly intuitively computable.

R1	$O(x) = 0$	*Zero function*
R2	$S(x) = x + 1$	*Successor function*
R3	$P_i^n(x_1, \dots, x_n) = x_i$	*i-th Projection*

By **substituting** functions into already given functions we obtain a new function:

R4	$f(\vec{x}) \simeq h(g_1(\vec{x}), \dots, g_k(\vec{x}))$	*Substitution*

Another important way to form new functions is by **primitive recursion**. According to this pattern the arithmetic operations addition, multiplication and exponentiation are defined.

R5	$f(\vec{x}, 0) \simeq g(\vec{x})$ $f(\vec{x}, y+1) \simeq h(\vec{x}, y, f(\vec{x}, y))$	*Primitive recursion*

Definition 1

The class **PRIM** of all **primitive recursive** (abbreviated: **p.r.**) functions is the smallest class of functions that contains the initial functions and is closed under substitution and primitive recursion. These functions are in particular total functions.

Examples of Primitive Recursive Functions

The following functions are primitive recursive:

$x + y$	*Sum*
$x \cdot y$	*Product*
x^y	*Exponentiation*
$\max(x, y)$	*Maximum*
$\min(x, y)$	*Minimum*
$\lvert x - y \rvert$	*Absolute difference*

and moreover,

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y, \\ 0 & \text{otherwise} \end{cases} \qquad \textit{Monus (difference on } \mathbb{N}\textit{)}$$

$$sg(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0 \end{cases} \qquad \textit{Sign function}$$

$$\overline{sg}(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{if } x > 0 \end{cases} \qquad \textit{Negated sign function}$$

Proof. Addition, multiplication and exponentiation are obviously primitive recursive. For the remaining cases one first shows that the predecessor function $x \dot{-} 1$ is primitive recursive, namely because

$$\begin{aligned} 0 \dot{-} 1 &= 0 \\ (x+1) \dot{-} 1 &= x \end{aligned}$$

and then defines $\dot{-}$ by primitive recursion

$$\begin{aligned} x \dot{-} 0 &= x \\ x \dot{-} (y+1) &= (x \dot{-} y) \dot{-} 1 \end{aligned}$$

The remaining functions can be defined directly using $+$ and $\dot{-}$:

$$\begin{aligned} |x - y| &= (x \dot{-} y) + (y \dot{-} x) \\ \max(x, y) &= x + (y \dot{-} x) \\ \min(x, y) &= \max(x, y) \dot{-} |x - y| \\ \overline{sg}(x) &= 1 \dot{-} x \\ sg(x) &= 1 \dot{-} \overline{sg}(x) \end{aligned}$$

□

Closure Properties

Exercise 1

Show that primitive recursive functions are closed under
(i) Case distinction:
If $g, f_0, f_1, \ldots, f_k$ are primitive recursive, then so is f with

$$f(\vec{x}) = \begin{cases} f_0(\vec{x}) & \text{if} \quad g(\vec{x}) = 0, \\ f_1(\vec{x}) & \text{if} \quad g(\vec{x}) = 1, \\ \vdots & \vdots \\ f_k(\vec{x}) & \text{if} \quad g(\vec{x}) \geq k. \end{cases}$$

and
(ii) Bounded μ-operator:
If g is primitive recursive, then so is f with

$$f(\vec{x}, z) = \mu y < z \; (g(\vec{x}, y) = 0),$$

where

$$\mu y < z \; R(\vec{x}, y) = \begin{cases} \text{the smallest } y < z \text{ with } R(\vec{x}, y) & \text{if such } y \text{ exists} \\ z & \text{otherwise.} \end{cases}$$

Recursive and Partially Recursive Functions

An example of a computable but not primitive recursive function is the **Ackermann function**. This is a 2-ary function that is recursively defined by the following equations:

$$\begin{aligned} A(0,y) &= y+1 \\ A(x+1,0) &= A(x,1) \\ A(x+1,y+1) &= A(x,A(x+1,y)) \end{aligned}$$

This is a double recursion, yet one can easily verify that each value $A(x,y)$ is determined by finitely many "earlier" values $A(u,v)$ with $u<x$ or $u=x \wedge v<y$. Writing $F_n(m)$ for $A(n,m)$, we obtain:

$$\begin{aligned} F_0(m) &= m+1 \\ F_1(m) &= m+2 \\ F_2(m) &\approx 2m \\ F_3(m) &\approx 2^m \\ F_4(m) &\approx 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \quad (m \text{ times}) \end{aligned}$$

This means that the Ackermann function iterates the basic arithmetic operations and grows very rapidly, particularly in the first argument (which indicates the number of iterations). In fact, it majorizes every primitive recursive function: If f is a k-ary p.r. function, then there exists a number n with

$$f(x_1,\dots,x_k) \leq F_n(\max(x_1,\dots,x_k))$$

for all $x_1,\dots,x_k$. Thus the 2-ary function A *cannot* be p.r. (although on the other hand all F_n are primitive recursive). That the Ackermann function is nevertheless computable can be seen from the fact that it can be obtained from a p.r. function by means of **minimalization**:

R6	$f(\vec{x}) \simeq \mu y\,(g(\vec{x},y) \simeq 0)$	μ-operator

that is,

$$f(\vec{x}) = \begin{cases} \text{the smallest } y \text{ such that} & \text{if such a } y \text{ exists,} \\ \quad (i)\ g(\vec{x},z) \text{ is defined for all } z \leq y \text{ and} & \\ \quad (ii)\ g(\vec{x},y) = 0, & \\ \text{undefined} & \text{otherwise.} \end{cases}$$

This principle can lead from total to partial functions (e.g., for $g(x,y) = |x-y^2|$, as a function over $\mathbb{N} \times \mathbb{N}$).

Definition 2

The class **R** of all **recursive functions** is the smallest class of functions that contains the initial functions and is closed under substitution, primitive recursion and minimalization. These functions are often also referred to as *general recursive*, *μ-recursive* functions or (based on the Church-Turing thesis) as *computable functions*. We will use *recursive* functions primarily to denote total (i.e. everywhere defined) computable functions. If the functions involved are partial, we will use *partial recursive* and *partial computable*.

Thus every primitive recursive function is also a recursive function (while the Ackermann function is an example of a recursive, but not p.r. function), and the closure properties from previous sections (suitably modified for partial functions) also hold for recursive functions.

Recursive Relations

Definition 3

A number-theoretic relation $R \subseteq \mathbb{N}^k$ is called **(primitive) recursive** if and only if its characteristic function c_R is (primitive) recursive, where

$$c_R(\vec{x}) = \begin{cases} 1 & \text{if } R(\vec{x}), \\ 0 & \text{otherwise.} \end{cases}$$

Thus, for example, the predicates $x = y$, $x \neq y$, $x \leq y$, $x < y$ are primitive recursive. Recursive relations are also called **decidable**.

Definition 4

A relation $R \subseteq \mathbb{N}^k$ is called **recursively enumerable** (*r.e.*), often also *computably enumerable (c.e.)* if it is the domain of a partial computable function:

$$R \text{ r.e.} : \Longleftrightarrow \; R = \{\vec{x} \mid f(\vec{x}) \downarrow\} \;\text{ for a computable function }\; f.$$

Some Remarks

- A non-empty set $A \subseteq \mathbb{N}^k$ is r.e. if and only if A is the range of a total computable function.
- If R is a decidable relation, then there is an effective procedure that determines if R holds or not. An r.e. set is also called **semi-decidable**, since in this case we have a procedure to list the elements.
- If both a relation and its complement are r.e., we can run both listing procedures in parallel and wait until an element gets listed. Thus

  $$A \text{ decidable} \quad \Leftrightarrow \quad A \text{ and } \mathbb{N}^k \setminus A \text{ are r.e.}$$

- We will later see that if T is a theory consisting of a computable set of sentences, its deductive closure

  $$\{\sigma : T \vdash \sigma\}$$

  is r.e.. If T is complete, this will imply that the deductive closure is decidable.